

2/PCT

10/534346  
JC06 Rec'd PCT/PTO 09 MAY 2005

WO 2004/045162

PCT/GB2003/004893

- 1 -

## TRAFFIC MANAGEMENT ARCHITECTURE

### Field of the Invention

- 5 The present invention concerns the management of traffic, such as data and communications traffic, and provides an architecture for a traffic manager that surpasses known traffic management schemes in terms of speed, efficiency and reliability.

### Background to the Invention

- 10 The problem that modern traffic management schemes have to contend with is the sheer volume. Data arrives at a traffic handler from multiple sources at unknown rates and volumes and has to be received, sorted and passed on "on the fly" to the next items of handling downstream. Received data may be associated with a number of attributes by which priority allocation, for example, is applied to individual data packets or streams, depending on the class of service offered to an individual client. Some traffic may therefore have to be queued whilst later arriving but higher priority traffic is processed.
- 15 A router's switch fabric can deliver packets from multiple ingress ports to one of a number of egress ports. The linecard connected to this egress port must then transmit these packets over some communication medium to the next router in the network. The rate of transmission is normally limited to a standard rate. For instance, an OC-768 link would transmit packets over an optical fibre at a rate of 40 Gbits/s.
- 20 With many independent ingress paths delivering packets for transmission at egress, the time-averaged rate of delivery cannot exceed 40 Gbits/s for this example. Although over time the input and output rates are equivalent, the short term delivery of traffic by the fabric is "bursty" in nature with rates often peaking above the 40 Gbits/s threshold. Since the rate of receipt can be greater than the rate of transmission, short term packet queueing
- 25 is required at egress to prevent packet loss. A simple FIFO queue is adequate for this purpose for routers which provide a flat grade of service to all packets. However, more complex schemes are required in routers which provide Traffic Management. In a converged internetwork, different end user applications require different grades of service in order to run effectively. Email can be carried on a best effort service where no
- 30 guarantees are made regarding rate of or delay in delivery. Real-time voice data has a much more demanding requirement for reserved transmission bandwidth and guaranteed minimum delay in delivery. This cannot be achieved if all traffic is buffered in the same

BEST AVAILABLE COPY

FIFO queue. A queue per so-called "Class of Service" is required so that traffic routed through higher priority queues can bypass that in lower priority queues. Certain queues may also be assured a guaranteed portion of the available output line bandwidth.

On first sight the traffic handling task appears to be straightforward. Packets are placed in queues according to their required class of service. For every forwarding treatment that a system provides, a queue must be implemented. These queues are then managed by the following mechanisms:

- Queue management assigns buffer space to queues and prevents overflow
- Measures are implemented to cause traffic sources to slow their transmission rates if queues become backlogged
- Scheduling controls the de-queuing process by dividing the available output line bandwidth between the queues.

Different service levels can be provided by weighting the amount of bandwidth and buffer space allocated to different queues, and by prioritised packet dropping in times of congestion. Weighted Fair Queueing (WFQ), Deficit Round Robin (DRR) scheduling, Weighted Random Early Detect (WRED) are just a few of the many algorithms which might be employed to perform these scheduling and congestion avoidance tasks.

In reality, system realisation is confounded by some difficult implementation issues:

- High line speeds can cause large packet backlogs to rapidly develop during brief congestion events. Large memories of the order 500 MBytes to 1 GBytes are required for 40 Gbits/s line rates.

- The packet arrival rate can be very high due to overspeed in the packet delivery from the switch fabric. This demands high data read and write bandwidth into memory. More importantly, high address bandwidth is also required.
- The processing overhead of some scheduling and congestion avoidance algorithms is high.

Priority queue ordering for some (FQ) scheduling algorithms is a non-trivial problem at high speeds.

- A considerable volume of state must be maintained in support of scheduling and congestion avoidance algorithms, to which low latency access is required. The volume of state increases with the number of queues implemented.

• As new standards and algorithms emerge, the specification is a moving target. To find a flexible (ideally programmable) solution is therefore a high priority.

In a conventional approach to traffic scheduling, one might typically place packets directly into an appropriate queue on arrival, and then subsequently dequeue packets from

5 those queues into an output stream.

Figure 1 shows the basic layout of the current approach to traffic management. It can be thought of as a "queue first, think later" strategy. Data received at the input 1 is split into a number of queues in parallel channels 2.1 to 2.n. A traffic scheduler processor 3 receives

10 the data from the parallel channels and sorts them into order. The order may be determined by the priority attributes, for example, mentioned above. State is stored in memory 4 accessible by the processor. The output from the processor represents the new order as determined by the processor in dependence on the quality of service attributes assigned to the data at the outset.

The traffic scheduler 3 determines the order of de-queuing. Since the scheduling decision  
15 can be processing-intensive as the number of input queues increases, queues are often arranged into small groups which are locally scheduled into an intermediate output queue. This output queue is then the input queue to a following scheduling stage. The scheduling problem is thus simplified using a "divide-and-conquer" approach, whereby high performance can be achieved through parallelism between groups of queues in a tree type

20 structure, or so-called hierarchical link sharing scheme.

This approach works in hardware up to a point. For the exceptionally large numbers of input queues (of the order 64k) required for per-flow traffic handling, the first stage becomes unmanageably wide to a point that it becomes impractical to implement the required number of schedulers.

25 Alternatively, in systems which aggregate all traffic into a small number of queues parallelism between hardware schedulers cannot be exploited. It then becomes extremely difficult to implement a single scheduler - even in optimised hardware - that can meet the required performance point.

30 With other congestion avoidance and queue management tasks to perform in addition to scheduling, it is apparent that a new approach to traffic handling is required. The queue first, think later strategy often fails and data simply has to be jettisoned. There is

therefore a need for an approach to traffic management that does not suffer from the same defects as the prior art and does not introduce its own fallibilities.

#### Summary of the invention

In one aspect, the invention provides a system comprising means for sorting incoming

- 5 data packets in real time before said packets are stored in memory.
- In another aspect, the invention provides a data packet handling system, comprising means whereby incoming data packets are assigned an exit order before being stored in memory.

- 10 In yet another aspect, the invention provides a method for sorting incoming data packets in real time, comprising sorting the packets into an exit order before storing them in memory.

The sorting means may be responsive to information contained within a packet and/or within a table and/or information associated with a data packet stream in which said packet is located, whereby to determine an exit order number for that packet.

- 15 The packets may be inserted into one or more queues by a queue manager adapted to insert packets into the queue means in exit order. There may be means to drop certain packets before being output from said queue means or before being queued in the queue means.

- 20 The system may be such that the sorting means and the queue means process only packet records containing information about the packets, whereas data portions of the packets are stored in the memory for output in accordance with an exit order determined for the corresponding packet record.

The sorting means preferably comprises a parallel processor, such as an array processor, more preferably a SIMD processor.

- 25 There may be further means to provide access for the parallel processors to shared state. A state engine may control access to the shared state.

- Tables of information for sorting said packets or said packet records may be provided, wherein said tables are stored locally to each processor or to each processor element of a parallel processor. The tables may be the same on each processor or on each processor
- 30 element of a parallel processor. The tables may be different on different processors or on different processor elements of a parallel processor.

The processors or processor elements may share information from their respective tables,

such that: (a) the information held in the table for one processor is directly accessible by a different processor or the information held in the table in one processor element may be accessible by other processing element(s) of the processor; and (b) processors may have access to tables in other processors or processor elements have access to other processor elements in the processor, whereby processors or processor elements can perform table lookups on behalf of other processor(s) or processor elements of the processor.

The invention also encompasses a computer system, comprising a data handling system as previously specified; a network processing system, comprising a data handling system as previously specified; and a data carrier containing program means adapted to perform a corresponding method.

#### **Brief Description of the Drawings**

The invention will be described with reference to the following drawings, in which:

Figure 1 is a schematic representation of a prior art traffic handler; and

Figure 2 is a schematic representation of a traffic handler in accordance with the invention.

#### **Detailed Description of the Illustrated Embodiments**

The present invention turns current thinking on its head. Figure 2 shows schematically the basic structure underlying the new strategy for effective traffic management. It could be described as a "think first, queue later <sup>TM</sup>" strategy.

Packet data (traffic) received at the input 20 has the header portions stripped off and record portions of fixed length generated therefrom, containing information about the data, so that the record portions and the data portions can be handled separately. Thus, the data portions take the lower path and are stored in Memory Hub 21. At this stage, no attempt is made to organise the data portions in any particular order. However, the record portions are passed to a processor 22, such as a SIMD parallel processor, comprising one or more arrays of processor elements (PEs). Typically, each PE contains its own processor unit, local memory and register(s).

In contrast to the prior architecture outlined in Figure 1, the present architecture shares state 23 in the PE arrays under the control of a State Engine (not shown) communicating with the PE array(s). It should be emphasised that only the record portions are processed in the PE array. The record portions are all the same length, so their handling is predictable, at least in terms of length.

The record portions are handled in the processor 22. Here, information about the incoming packets is distributed amongst the PEs in the array. This array basically performs the same function as the processor 3 in the prior art (Figure 1) but the operations are spread over the PE array for vastly more rapid processing. This processing effectively

5 "time-stamps" the packet records to indicate when the corresponding data should be exited, assuming that it should actually be exited and not jettisoned, for example. The results of this processing are sent to the orderlist manager 24, which is an "intelligent" queue system which places the record portions in the appropriate exit order, for example in bins allocated to groups of data exit order numbers. The manager 24 is

10 preferably dynamic, so that new data packets with exit numbers having a higher priority than those already in an appropriate exit number bin can take over the position previously allocated. It should be noted that the PE array 22 simply calculates the order in which the data portions are to be output but the record portions themselves do not have to be put in that order. In other words, the PEs do not have to maintain the order of packets being

15 processed nor sort them before they are queued. Previous systems in which header and data portions were treated as one entity became unwieldy, slow and cumbersome because of the innate difficulty of preserving the integrity of the whole packet yet still providing enough bandwidth to handle the combination. In the present invention, it is only necessary for the Memory Hub 21 to

20 provide sufficient bandwidth to handle just the data portions. The memory hub can handle packets streaming in at real time. The memory hub can nevertheless divide larger data portions into fragments, if necessary, and store them in physically different locations, provided, of course, there are pointers to the different fragments to ensure read out of the entire content of such data packets.

25 In order to overcome the problem of sharing state over all the PEs in the array, multiple PEs are permitted to access (and modify) the state variables. Such access is under the control of a State Engine (not shown), which automatically handles the "serialisation" problem of parallel access to shared state.

The output 25, in dependence on the exit order queue held in the Orderlist Manager 24,

30 instructs the Memory Hub 21 to read out the corresponding packets in that required order, thereby releasing memory locations for newly received data packets in the process.

**BEST AVAILABLE COPY**

The chain-dotted line 26 enclosing the PE array 22, shared state/State Engine 23 and Orderlist Manager 24 signifies that this combination of elements can be placed on a single chip and that this chip can be replicated, so that there may be one or two (or more) chips interfacing with single input 20, output 25 and Memory Hub 21. As is customary, the  
5 chip will also include necessary additional components, such as a distributor and a collector per PE array to distribute data to the individual PEs and to collect processed data from the PEs, plus semaphore block(s) and interface elements.

The following features are significant to the new architecture:

- There are no separate, physical stage one input queues.
- 10 • Packets are effectively sorted directly into the output queue on arrival. A group of input queues thus exists in the sense of being interleaved together within the single output queue.
- These interleaved "input queues" are represented by state in the queue state engine. This state may track queue occupancy, finish time/number of the last packet in the queue  
15 etc. Occupancy can be used to determine whether or not a newly arrived packet should be placed in the output queue or whether it should be dropped (congestion management). Finish numbers are used to preserve the order of the "input queues" within the output queue and determine an appropriate position in the output queue for newly arrived packets (scheduling).
- 20 • Scheduling and congestion avoidance decisions are thus made "on the fly" prior to enqueueing (ie "Think first, queue later"<sup>TM</sup>).
- This technique is made possible by the deployment of a high performance data flow processor which can perform the required functions at wire speed. Applicant's array processor is ideal for this purpose, providing a large number of processing cycles per  
25 packet for packets arriving at rates as high as one every couple of system clock cycles.

#### Ancillary features

##### *Class of Service (CoS) tables:*

CoS parameters are used in scheduling and congestion avoidance calculations. They are conventionally read by processors as a fixed group of values from a class of service table  
30 in a shared memory. This places further demands on system bus and memory access bandwidth. The table size also limits the number of different classes of service which may be stored.

An intrinsic capability of Applicant's array processor is rapid, parallel local memory access. This can be used to advantage as follows:

- The Class of Service table is mapped into each PE's memory. This means that all passive state does not require lookup from external memory. The enormous internal memory addressing bandwidth of SIMD processor is utilised.
- By performing multiple lookups into local memories in a massively parallel fashion instead of single large lookups from a shared external table there is a huge number of different Class of Service combinations available from a relatively small volume of memory.
- Table sharing between PEs - PEs can perform proxy lookups on behalf of each other. A single CoS table can therefore be split across two PEs, thus halving the memory requirement.

#### Summary

- It can thus be appreciated that the present invention is capable of providing the following key features, marking considerable improvements over the prior art:
- Traditional packet scheduling involves parallel enqueueing and then serialised scheduling from those queues. For high performance traffic handling we have turned this around. Arriving packets are first processed in parallel and subsequently enqueued in a serial orderlist. This is referred to as "Think First Queue Later"<sup>TM</sup>
  - The deployment of a single pipeline parallel processing architecture (Applicant's array processor) is innovative in a Traffic Handling application. It provides the wire speed processing capability which is essential for the implementation of this concept.
  - An alternate form of parallelism (compared to independent parallel schedulers) is thus exploited in order to solve the processing issues in high speed Traffic Handling.